https://ntrs.nasa.gov/search.jsp?R=19950008173 2020-06-16T10:21:52+00:00Z

N95- 14587

18

COMPARISON OF TRANSFORM CODING METHODS WITH AN OPTIMAL PREDICTOR FOR THE DATA COMPRESSION OF DIGITAL ELEVATION MODELS

M. Lewis.
Department of Mathematics and Computing, The University of Glamorgan,
Pontypridd, Mid-Glamorgan, CF37 1DL., U.K.

Tel.U.K. 443 480480 Extn. 2714, Fax: U.K. 443 482711.
Email : MLEWIS@uk.ac.glam

ABSTRACT

Statistical encoding techniques enable the reduction of the number of bits required to encode a set of symbols, and are derived from their probabilities. Huffman encoding [1] is an example of statistical encoding that has been used for error-free data compression. The degree of compression given by Huffman encoding in this application can be improved by the use of prediction methods. These replace the set of elevations by a set of corrections that have a more advantageous probability distribution. In particular, the method of Lagrange Multipliers for minimisation of the mean square error has been applied to local geometrical predictors [3]. Using this technique, an 8-point predictor achieved about a 7% improvement over an existing simple triangular predictor [2].

In this paper, comparisons have been made between this predictive encoding methods and a transform coding technique, the Two-Dimensional Discrete Cosine Transform (2D-DCT). Transform coding allows greater compression but is computationally intensive and is subject to a greater degree of error on reconstruction of the data The Discrete Cosine Transform coding method can be combined with either Huffman encoding or Run Length Encoding (RLE) of the DCT coefficients to achieve greater compression. The method of blocking the DCT coefficients before Huffman encoding gives a better performance than Run Length encoding of the DCT coefficients The best compression achievable for the same data set using the slow DCT algorithm with blocking is about 35.24:1, i.e. a storage saving of 96.49% for at most an error of 5 metres and a root mean square error (rmse) of 0.5. For error-free compression (accurate to the nearest metre), the simple prediction method [2] gives a compression ratio of 13.04:1 with blocking the prediction errors before Huffman encoding. This gives a storage saving of about 92.30%. Similar results for a second more variable data set give a compression ratio of 17.60:1 or a storage saving of 94.12%, again for an equivalent maximum error of about 5 metres and an rmse of about 0.8. In the error-free Huffman method with blocking, equivalent results for the second data set are a compression ratio of about 7.13:1 and a storage saving of 85.93%. The Lagrange Multiplier method [3] will give an improvement of about 7% to the error-free compression ratios quoted. Since both these algorithms are computationally expensive, a trade-off between

maximum compression ratio and speed of compression/decompression must be made.

The use of another transform technique, the two dimensional Daubechies Wavelet transforms [4,5] shows similar performances with further blocking of the coefficients before Huffman encoding. The best performance for the first data set was a storage saving of 96.04% or a compression ratio of about 30:1 with the 12-coefficient ('smooth') transform with at most an error of 3 metres and a root mean square error (rmse) of 0.5. For the second more variable data set, a storage saving of 92.33% or a compression ratio of about 13:1 with the 4-coefficient ('local') transform. Here the maximum error is 3 metres with an rmse of 0.8. Further evaluation with other wavelet transforms is being studied as well as improved preprocessing of data to improve predictor efficiencies.

1. Data Compression in Digital Terrain Models.

The main emphasis in this work has not been on the encoding methods themselves but on the prediction methods specific to terrain that allow the coding methods to work better. Two transformation methods and one statistical encoding method have been chosen to apply to DEM's. These methods have been chosen because they have the potential to form robust data compression schemes with both good compression performance and moderate to good computational requirements.

In general, an estimate of the maximum amount of compression achievable in an error-free encoding process can be made by dividing the average number of bits needed to represent each terrain height in the original source data by a first-order estimate of the entropy of the prediction error data. Since there is in general a large degree of redundancy in the source data, the prediction process enables a reduction in the entropy value through this mapping process due to the probability density function of the prediction errors being highly peaked at zero and characterised by a relatively small variance.

In order to reduce the overall amount of data needed for storage, a prediction algorithm is employed. Since there is a close correlation between adjacent height values in a DTM, the differences between the actual and predicted values can be represented by fewer bits than the original data. These differences between the predictions and the actual elevations are recorded and Huffman encoded. Some base elevations such as two known axes of elevations are also stored.

One approach to the error-free compression of digital elevation data (DEM) involves the use of an identical predictor for both encoding and decoding processes. A terrain surface is normally considered to be a two-dimensional array representation of height values. Another approach to the design of an optimum predictor proved this triangular predictor to be sub-optimal and a better predictor

was devised using the method of Lagrange Multipliers [3]. ST06 and ST08 are two tiles taken from the Ordnance Survey regular 401x401 square grid with terrain heights accurate to the nearest metre. The former tile contains sea, coastal cliffs and relatively smooth changes in contours whilst the latter contains rougher terrain with deep valleys with large changes in contour values. A small improvement can be made to the simple triangular predictor method for both the three-point and eight-point predictors by the minimisation method of Lagrange multipliers. For many data sets compression ratios above 4 or 5 are easily achievable using a error-free Huffman encoding algorithm with minor modification to the code given in [2].

2.0 Application of a Transformation Method to a Digital Elevation Model.

2.1. Performance of the Two Dimensional Discrete Cosine Transform (2D-DCT).

The prediction/Huffman method described above is suitable for error-free encoding. The DCT method normally gives some error even without explicit quantisation. This is due to the representation of the real coefficients as integers. If the method were modified to make it error-free, it would give no compression. As well as evaluating various DCT methods, it is interesting to compare the DCT and prediction methods when error-free compression is not required. The prediction/Huffman method can be used in a lossy way whereby elevations are grouped into bands. e.g.

Band	Elevations (m)	Representative Elevation (m)
0	0	0.0
1	1 2 3 4 5	3.0
2	6 7 8 9 10	8.0
3	11 12 13 14 15	13.0

with a maximum error of 2.0 metres.

The bands are then used for prediction and correction. The maximum errors are small but the root mean square error (rmse) may be relatively large as most of the elevations may be in error. Note is also made that in some of the results in this section, the elevations have been divided by 2 before encoding , whereas in others the original elevations are used. In general, the entropy is smaller if the elevations have been divided by 2.

Tables 1A and 1B illustrate the effect of applying the Huffman encoding algorithm

to ST06 and ST08 after using the triangular prediction algorithm [2] when banding is used. A further improvement of up to 7% is possible by the use of the Lagrange Multiplier method [3]. It is evident that the blocking method is successful in reducing the average code length when the code efficiency is otherwise low.

2.2. Comparative Results using the Two Dimensional Discrete Cosine Transform Algorithm with Blocking for Huffman Encoding.

Discrete Cosine Transform techniques are data independent and samples in the transform domain are selected, quantised and coded according to the number of bits needed for compression. Here acceptable results are obtained for a wide range of compression ratios. Several studies were made using the Discrete Cosine Transform. Various subgrids were selected from ST06 and ST08 typically of size 16,32 and 64 square and the 2D-DCT and its inverse applied to increasingly varied terrain topography. The reconstructed terrain was compared graphically with the original. Further compression can be achieved by Huffman encoding the reduced set of coefficients. The errors introduced on decompression by the application of the inverse transform were variable with the largest range in height error appearing with the greatest compression ratio. Moreover, there was no visible detectable structure or linear relationship in the reconstructed errors but the source of the greatest error was associated with sharp changes in terrain profile i.e. valley sides and coastal cliff areas.

The 2D-DCT was encoded into an Ada program written specially for this study. The results presented here are for the case when 8x8 cells are used, but other sizes can be used. Quantisation values are not used, and relative coding of DC coefficients is optional. The algorithm is also combined with Huffman encoding. The effect of blocking the DCT coefficients prior to Huffman encoding was investigated. Again the data sets used were ST06 and ST08 and the DCT operation typically produces accuracy errors when the coefficients are converted to integers before Huffman encoding and back to floating point numbers before the inverse DCT transform is applied. The Huffman encoding of coefficients itself is error-free and reversible. These accuracy errors are duly noted as is the effect of banding the coefficients before Huffman encoding into sets of size 1-4. All the results in Table 2 are from applying these two algorithms to the original terrain height data using a window of size 8x8 'pixels'. In Table 3, the methods were applied to terrain data values firstly divided by 2 to reduce the range of coefficient values produced by the DCT transform for further comparative analysis with Tables 1A and 1B. Table 4 again shows equivalent results only this time 'differencing' was applied to the DCT term in each 8x8 window along each row-block and relative to the (0,0) position. Interestingly this worked well for ST08 but not for ST06. This was due to the overall reduction in scale for the DC coefficient values for ST08 that didn't affect ST06 since ST06 already had many more zero valued coefficients. This was based on the large number of zero values for sea areas in the original terrain.

2.3. Discussion on Efficiency Comparison.

In general combining these algorithms with blocking (Tables 2 &3) produced the best results. Since we are looking at code efficiencies close to 1, blocking the symbols overcomes the decrease in code efficiency when the entropy is less than 1 bit per symbol. In ST06, the algorithms (without 'differencing') produced the lowest entropy and average code length of 0.4540 and 0.5620 of a bit per coefficient (and hence bit per elevation) respectively. This is equivalent to a storage saving of 96.49% with a decompression error due to storage accuracy of the DCT coefficients of ±5 metres, a mean of 0.0011, an rmse of 0.5210 and a standard deviation of 0.5210 (with MINITAB™). Furthermore, one can compare with the prediction plus Huffman method of Tables 1A & 1B which have error values of ±6 metres, a mean of 1.0178, an rmse of 2.4042 and a standard deviation of 2.1715. This corresponds to an entropy value of 0.5013, an average code length of 0.5507 bits per elevation and a storage saving of 96.56%. The error distribution for the former DCT hybrid methods tend to be gaussian in form with the peak becoming more rounded as the coefficients are grouped into larger sets. In the latter prediction hybrid methods to include errors, the error profile can be said to be 'flat' in form. In ST08, the best performance for comparative measures comes from Table 4. Using the DCT method and a blocksize of 4 for Huffman encoding of the coefficients, an entropy of 0.9091 and an average code length of 0.9409 bit per coefficient (bit per elevation) corresponds to a storage saving of 94.12%. Noted is the fact that the efficiency of the Huffman encoding algorithm decreases when the block size is greater than 3 for ST08. The error profile gives a maximum error range of about 5 metres, a mean of 0.003, an rmse of 0.7898 and a standard deviation of 0.7893 (see below). Here, for comparative results we must look at Table 4B and an error range of ± 4 (actual 8) metres. In this case, blocking by 4 to give an entropy of 0.9383 bit per elevation and an average code length of 0.9450 bits per elevation achieves a storage saving of 94.09%. However, the overall errors are much bigger: the rmse is 2.5974 and the standard deviation is 2.5909.

The conclusion to be drawn from these results is that for lossy compression, the DCT/Huffman method gives the best compromise between compression and error, but only when blocking takes place before Huffman encoding. This may however, create a computational overhead that makes the method unattractive in practice.

Huffman Encoding with Allowable Errors, using Error Banding and the Triangular Prediction Algorithm [3].
(Heights/ 2)

ST06

Error Range ± m (Rmse)	Blocksize	Entropy (bits /elevation)	Average Code Length (bits /elevation)	Code Efficiency (%)	Storage Saving (%)
1	1	1.3910	1.5642	88.9304	90.2239
	2	1.3273	1.3443	98.7376	91.5984
(0)	3	1.2760	1.2807	99.6340	91.9957
	4	1.2268	1.2313	99.6328	92.3041
2	1	0.8914	1.2789	69.7018	92.0069
	2	0.8418	0.9251	90.9994	94.2183
(0.8904)	3	0.8059	0.8334	96.6923	94.7910
	4	0.7823	0.7938	98.5473	95.0387
4	1	0.7052	1.2003	58.7541	92.4980
	2	0.6613	0.7983	82.8448	95.0108
(1.6860)	3	0.6319	0.6888	91.7354	95.6947
	4	0.6114	0.6411	95.3641	95.9931
6	1	0.5809	1.1535	50.3590	92.7905
	2	0.5434	0.7291	74.5297	95.4433
(2.4042)	3	0.5167	0.6034	85.6364	96.2287
	4	0.5013	0.5507	91.0170	96.5579
8	1	0.4936	1.1231	43.9531	92.9809
	2	0.4598	0.6840	67.2177	95.7250
(3.1391)	3	0.4361	0.5499	79.3044	96.5632
	4	0.4236	0.4918	86.1303	96.9260
10	1	0.4186	1.0991	38.0836	93.1303
	2	0.3861	0.6473	59.6486	95.9545
(3.9149)	3	0.3694	0.5072	72.8387	96.8301
	4	0.3551	0.4419	80.3544	97.2380

Table 1A.

Huffman Encoding with Allowable Errors, using Error Banding and the Triangular Prediction Algorithm [3].
(Heights / 2)

ST08

Error Range ± m (Rmse)	Blocksize	Entropy (bits /elevation)	Average Code Length (bits /elevation)	Code Efficiency (%)	Storage Saving (%)
1	1	2.3689	2.4386	97.1401	84.7585
	2	2.3376	2.3563	99.2034	85.2730
(0)	3	2.3043	2.3133	99.6112	85.5416
	4	2.2444	2.2513	99.6947	85.9293
2	1	1.5201	1.6559	91.8011	89.6509
	2	1.4780	1.5084	97.9854	90.5724
(0.8173)	3	1.4531	1.4621	99.3861	90.8619
	4	1.4342	1.4435	99.3570	90.9784
4	1	1.3074	1.5187	86.0840	90.5082
	2	1.2456	1.2700	98.0804	92.0626
(1.4549)	3	1.2139	1.2364	98.1864	92.2728
	4	1.1900	1.1963	99.4770	92.5234
6	1	1.1697	1.4334	81.6063	91.0412
	2	1.1012	1.1316	97.3092	92.9273
(2.0005)	3	1.0664	1.0758	99.1217	93.2762
	4	1.0394	1.0548	98.5455	93.4077
8	1	1.0667	1.3738	77.6409	91.4136
	2	1.0015	1.0440	95.9312	93.4751
(2.5974)	3	0.9649	0.9709	99.3886	93.9322
	4	0.9383	0.9450	99.2922	94.0937
10	1	0.9742	1.3247	73.5369	91.7204
	2	0.9121	0.9726	93.7827	93.9214
(3.9149)	3	0.8751	0.8841	98.9866	94.4744
	4	0.8500	0.8537	99.5640	94.6643

Table 1B.

Two Dimensional Discrete Cosine Transform (‡)with Huffman Encoding of Coefficients.
(Original Heights, 8x8 windows)

ST06

Block-size	Entropy (bits/ Coefficient)	Average Code Length (bits/ Coefficient)	Code Efficiency (%)	Storage Saving (%)
1	0.998178	1.4731	67.7598	90.7930
2	0.840046	1.0229	82.1276	93.6072
3	0.835856	0.9178	91.0756	94.2640
4	0.620969	0.6870	90.3936	95.7065

Error Range (metres) on Reconstruction through Rounding before Huffman Encoding:-

Max = 5.0000
Min = -5.1875
Mean = - 0.003
Rmse = 0.6716

ST08

Block-size	Entropy (bits/ Coefficient)	Average Code Length (bits/ Coefficient)	Code Efficiency (%)	Storage Saving (%)
1	2.069000	2.2048	93.8423	86.2200
2	1.775474	1.7967	98.8210	88.7709
3	1.742708	1.7478	99.7080	89.0762
4	1.248854	1.2528	99.6853	92.1700

Error Range (metres) on Reconstruction through Rounding before Huffman Encoding:-

Max = 4.7082
Min = -4.4846
Mean = - 0.0013
Rmse = 0.9726

Table 2.

Two Dimensional Discrete Cosine Transform (‡) with Huffman Encoding of Coefficients
(Heights/2, 8x8 Windows)

ST06

Block-size	Entropy (bits/ Coefficient)	Average Code Length (bits/ Coefficient)	Code Efficiency (%)	Storage Saving (%)
1	0.650300	1.2818	50.7309	91.9885
2	0.541816	0.8142	66.5469	94.9113
3	0.554087	0.6928	79.9790	95.6701
4	0.454014	0.5620	80.7887	96.4876
Error Range (metres) on Reconstruction through Rounding before Huffman Encoding:-			Max = 5.0939 Min = -5.0300 Mean = 0.0011 Rmse = 0.5210 148630 Zero Coefficients	

ST08

Block-size	Entropy (bits/ Coefficient)	Average Code Length (bits/ Coefficient)	Code Efficiency (%)	Storage Saving (%)
1	1.401504	1.7376	80.6574	89.1400
2	1.178912	1.2883	91.5101	91.9482
3	1.180569	1.2142	97.2263	92.4109
4	0.909866	0.9417	96.6206	94.1144
Error Range (metres) on Reconstruction through Rounding before Huffman Encoding:-			Max = 6.1186 Min = -4.5141 Mean = 0.0030 Rmse = 0.7898 132146 Zero Coefficients	
‡Typical Calculation Time for Slow Algorithm: Forward DCT 5m 1sec. , Inverse DCT 5m 46 secs.				

Table 3.

Two Dimensional Discrete Cosine Transform (‡) with Huffman Encoding of Coefficients .

(Heights/ 2, 8x8 Windows with DC Term Row Differencing)

ST06

Block-size	Entropy (bits/ Coefficient)	Average Code Length (bits/ Coefficient)	Code Efficiency (%)	Storage Saving (%)
1	0.6591	1.2874	51.1960	91.9539
2	0.5531	0.8226	67.2388	94.8587
3	0.5652	0.7017	80.5411	95.6142
4	0.4650	0.5708	81.4715	96.4327
Error Range (metres) on Reconstruction through Rounding before Huffman Encoding:-			Max = 5.0939 Min = -5.0300 Mean = 0.0011 Rmse = 0.5210 148496 Zero Coefficients	

ST08

Block-size	Entropy (bits/ Coefficient)	Average Code Length (bits/ Coefficient)	Code Efficiency (%)	Storage Saving (%)
1	1.3941	1.7303	80.5700	89.1856
2	1.1755	1.2848	91.4876	91.9697
3	1.1777	1.2114	97.2189	92.4290
4	0.9091	0.9409	96.6204	94.1194
Error Range (metres) on Reconstruction through Rounding before Huffman Encoding:-			Max = 6.1186 Min = -4.5141 Mean = 0.0030 Rmse = 0.7898 132146 Zero Coefficients	

‡Typical Calculation Time for Slow Algorithm:
Forward DCT 5m 1sec. , Inverse DCT 5m 46 secs.

Table 4.

3.0. The Use of Wavelet Transforms for the Data Compression of DEM's.

The key idea with wavelet transforms is in the formation of classes of signals into weighted sums of basis functions (complex exponentials for the Fourier Transform and cosines for the Cosine Transform). In contrast to traditional Fourier theory, the basis functions are formed by scaling and translating a single function and the mathematical properties of the decomposition are determined by the properties of the underlying function. Thus unlike sines and cosines, which define an unique Fourier or Cosine transform, there is not one single unique set of wavelets; in fact, there are infinitely many possible sets. A particular set of wavelets is specified by a particular set of numbers, called wavelet filter coefficients. One such simple set comes from a class discovered by Daubechies [4] which include members ranging from being highly localised to highly smooth. Press [5] describes both the transformation methods for the simple case and how the Discrete Wavelet Transform (DWT) is formalised. Compact (and therefore unsmooth) wavelets are better for lower accuracy approximation and for functions with discontinuities (like edges), while smooth (and therefore non-compact) wavelets are better for achieving high numerical accuracy. By taking a multi-dimensional wavelet transform of an image, compression is achieved by bit allocation amongst the coefficients in some highly non-uniform, optimised way. In general, large wavelet coefficients are quantised accurately, whilst small coefficients are quantised coarsely with only a bit or two or may even be truncated completely. If the resulting quantisation levels are still statistically non-uniform, they may then be further compressed by a technique such as Huffman encoding. When a smooth and coarser wavelet transform are applied to a DEM, the performance is much the same in terms of reconstruction error from accuracy and compression ratio. There is however a slight improvement in computation time.

Two Dimensional 4-Coefficient ('Localised') Daubechies Wavelet Transform with Huffman Encoding of Coefficients.

(Original Heights)

ST08

256x256 subset

Block-size	Entropy (bits/ Coefficient)	Average Code Length (bits/ Coefficient)	Code Efficiency (%)	Storage Saving (%)
1	3.3885	3.4170	99.1654	78.6438
2	3.0484	3.0666	99.4068	80.8335
3	2.6881	2.6994	99.5842	83.1290
4 •	1.8727	1.8807	99.5721	88.2456

Invertible Wavelet Transform.
Error Range (metres) on Reconstruction through **Rounding** before Huffman Encoding:-

Max = 3.0
Min = -2.0
Mean = 0.50
Rmse = 0.79
Stdev = 0.61

32093 'Zero' Coefficients, ST08 Coordinates (10,10).
Typical Transform Calculation time 20 secs.
• Typical computation time 3 mins (DEC Alpha).

Table 5.

Two Dimensional 12-Coefficient ('Smooth') Daubechies Wavelet Transform with Huffman Encoding of Coefficients.

(Original Heights/2)

ST06

256x256 subset

Block-size	Entropy (bits/ Coefficient)	Average Code Length (bits/ Coefficient)	Code Efficiency (%)	Storage Saving (%)
1	1.0261	1.5070	68.0902	90.5815
2	0.8619	1.0489	82.1721	93.4443
3	0.7425	0.8438	88.0017	94.7265
4 •	0.5383	0.6335	84.9665	96.0406

Invertible Wavelet Transform.
Error Range (metres) on Reconstruction through **Rounding** before Huffman Encoding:-

Max = 2.0
Min = -3.0
Mean = -0.26658
Rmse = 0.56420
Stdev = 0.49784

57829 'Zero' Coefficients, ST06 Coordinates (0,0).
Typical Transform Calculation time 20 secs.

• Typical computation time 3 mins (DEC Alpha)

Table 6.

Two Dimensional 12-Coefficient ('Smooth') Daubechies Wavelet Transform with Huffman Encoding of Coefficients.
(Original Heights/2)

ST06
256x256 subsets

Block-size	Entropy (Bits/ Coefficient)	Average Code Length (Bits/ Coefficient)	Code Efficiency (%)	Storage Saving (%)
Coordinates (0,0)				
1	1.0261	1.5070	68.0902	90.5815
2	0.8619	1.0489	82.1721	93.4443
3	0.7425	0.8438	88.0017	94.7265
4 •	0.5383	0.6335	84.9665	96.0406
Coordinates (0,144)				
1	0.9633	1.4743	65.3437	90.7859
2	0.8001	1.0047	79.6315	93.7203
3	0.6830	0.7983	85.5519	95.0103
4	0.4817	0.5833	82.5810	96.3542
Coordinates (144,0)				
1	1.3991	1.7360	80.5957	89.1500
2	1.1993	1.3067	91.7783	91.8329
3	1.0436	1.0935	95.4397	93.1658
4	0.7606	0.8132	93.5356	94.9174
Coordinates (144,144)				
1	1.3251	1.6937	78.2379	89.4145
2	1.1294	1.2551	89.9883	92.1556
3	0.9791	1.0402	94.1217	93.4985
4	0.7308	0.7924	92.2235	95.0475

Typical Transform Calculation time 20 secs.	• Typical Computation time 3mins. (DEC Alpha)
Invertible Wavelet Transform. Error Range (metres) on Reconstruction through **Rounding** before Huffman Encoding:-	Max = 2.0 Min = -3.0 Mean = -0.26658 Rmse = 0.56420 Stdev = 0.49784

Table 7.

Two Dimensional 4-Coefficient ('Localised') Daubechies Wavelet Transform with Huffman Encoding of Coefficients.
(Original Heights/2)

ST08
256x256 subsets

Block-size	Entropy (Bits/ Coefficient)	Average Code Length (Bits/ Coefficient)	Code Efficiency (%)	Storage Saving (%)
Coordinates (0,0)				
1	2.2938	2.4014	95.5188	84.9915
2	2.0289	2.0447	99.2277	87.2208
3	1.7923	1.8012	99.5079	88.7426
4 •	1.2202	1.2268	99.4639	92.3325
Coordinates (0,144)				
1	2.2078	2.3358	94.5171	85.4010
2	1.9512	1.9723	98.9299	87.6732
3	1.7220	1.7307	99.4990	89.1834
4 •	1.2285	1.2377	99.2570	92.2645
Coordinates (144,0)				
1	2.5926	2.6566	97.5900	83.3962
2	2.3031	2.3140	99.5265	85.5372
3	2.0367	2.0517	99.2679	87.1770
4 •	1.3939	1.3970	99.7746	91.2684
Coordinates (144,144)				
1	2.2902	2.4022	95.3366	84.9862
2	2.0291	2.0459	99.1780	87.2130
3	1.7932	1.8020	99.5095	88.7375
4 •	1.2568	1.2636	99.4646	92.1026

Typical Transform Calculation time 20 secs.	• Typical Computation time =3 mins (DEC Alpha)
Invertible Wavelet Transform. Error Range (metres) on Reconstruction through **Rounding** before Huffman Encoding:-	Max = 3.0 Min = -2.0 Mean = 0.50433 Rmse = 0.7879 Stdev = 0.60534

Table 8.

3.1. Results.

Both sets of test data were partitioned into subsets of 256x256 arrays and both the two-dimensional 'localised' 4-coefficient and the 'smooth' 12-coefficient Daubechies wavelet transform were applied in turn to each data set. In all cases the resulting coefficient matrices were Huffman encoded with statistical analyses made as described above. The only errors on reconstruction were due to the rounding of the coefficients before Huffman encoding leading to some loss of accuracy on reconstructing the original data.

Table 5 shows the results of applying the 'localised' Daubechies wavelet transform to a typical subset of ST08 using the original terrain heights in the data vector. This 'localised' transform performed better than the 'smoother' transforms (12 and 20 coefficient) on the rougher terrain data and the results for storage savings can therefore be compared to using the Two-Dimensional Discrete Cosine Transform (2D-DCT) in Table 2. This method together with blocking the coefficients before Huffman encoding achieves an average code length of 1.8807 bits per coefficient compared with 1.2528 bits per coefficient for the 2D-DCT case. This in turn gives the comparative storage savings of 88.25% and 92.17 respectively. The advantage that the wavelet method has over the DCT method is in the efficiency of the algorithm, it is much faster even when blocking is done. In the same light, Table 6 where the original heights of ST06 are divided by 2, can be compared to the results in the top half of Table 3. For ST06, the 'smoother' 12-coefficient wavelet transform worked best. Although, the matrices are of differing sizes, again the results are marginally inferior in terms of average code length but the final storage savings are about the same (96%). Tables 7 and 8 describe in full the results for the four 256x256 overlapping squares from ST06 and ST08 for original terrain data divided by 2, the smother wavelet working better on the smoother terrain of ST06 and the more localised wavelet best on the more varied landscape of ST08.

The results overall are marginally worse than with the 2D-DCT if one compares Tables 2 & 3 with Tables 6 or Table 7. Indeed when the accuracy errors are compared with Tables 4A and 4B where errors are introduced (through banding) into the data prior to the prediction algorithm and Huffman encoding, still gives the prediction method an advantage over the transformation methods. For example, in a typical subset of ST08, the 4-coefficient Daubechies wavelet transform with Huffman encoding of the coefficients (when the coefficients are blocked into sets of four), gives an entropy value of 1.8727 bits per coefficient and a storage saving of 88.25%. This is when the min./max. error is 3 and -2 metres and the rmse is 0.79 (Table 12). Table 4B shows the equivalent performance when the prediction algorithm and error banding is used for ST08. In this case for a max./min. error of 2 metres and an rmse of 0.8173, comparative results with blocking sets of prediction errors into sets of 4 gives an entropy of 1.4342 bits/elevation and a storage saving of 90.98%. Although the rmse values are about the same, the range of errors is much smaller.

4.0. Summary of Research.

Comparisons have been made between these predictive encoding methods and a transform coding technique, the Two-Dimensional Discrete Cosine Transform (2D-DCT). Transform coding allows greater compression but is computationally intensive and is subject to a greater degree of error on reconstruction of the data. The Discrete Cosine Transform coding method can be combined with either Huffman encoding or Run Length Encoding (RLE) of the DCT coefficients to achieve greater compression. Further compression can be achieved when using the Huffman DCT method by blocking the transformation coefficients. The method of blocking the DCT coefficients before Huffman encoding gives a better performance than Run Length encoding of the DCT coefficients The best compression achievable for one data set (ST06) using the DCT algorithm with blocking is about 35.24:1, i.e. a storage saving of 96.49% for at most an error of 5 metres and a root mean square error (rmse) of 0.5. For error-free compression (accurate to the nearest metre), the simple prediction method [2] gives a compression ratio of 13.04:1 with blocking the prediction errors before Huffman encoding. This gives a storage saving of about 92.30%. Similar results for a second more variable data set (ST08) using the DCT algorithm, give a compression ratio of 17.60:1 or a storage saving of 94.12%, again for an equivalent maximum error of about 5 metres and an rmse of about 0.8. In the error-free Huffman method with blocking, equivalent results for the second data set (ST08) are a compression ratio of about 7.13:1 and a storage saving of 85.93%. The wavelet transform method tested produces similar but marginally inferior results compared to the 2D-DCT. The Lagrange Multiplier method [3] will give an improvement of about 7% to the error-free compression ratios quoted. There remain a large number of possible variations on the method, including the use of arithmetic, adaptive Huffman or adaptive arithmetic coding in place of static Huffman encoding. Wavelet transforms give similar results to the DCT but are much more efficient. Since both these algorithms are computationally expensive, a trade-off between maximum compression ratio and speed of compression/decompression must be made.

Current work involves looking at different families of wavelets for compression and developing efficient heuristic algorithms to selectively constrain the terrain topography to improve prediction methods.

[1] Huffman D.A. "A Method for the Construction Of Minimum Redundancy Codes". Proc. IRE, Vol. 40, 1952. pp.1098-1101.

[2] Kidner D.B. & Smith D.H. "Compression of Digital Elevation Models by Huffman Encoding"; Computers & Geosciences, Vol. 18., No. 8, pp.1013-1034, 1992.

[3] Lewis M. & Smith D.H. "Optimal Predictors for the Data Compression of Digital

Elevation Models Using the Method of Lagrange Multipliers". pp. 246-256. Proc Auto Carto 11, Oct. 30-Nov 3rd 1993, Minneapolis, Minnesota,USA.

[4] Daubechies I., 1988 Commun. Pure & Applied Maths, vol.44, pp.141–183.

[5] Press W.H., "Wavelet Transforms", 2nd Edition of Numerical Recipes:The Art of Scientific Computing. Harvard–Smithsonian Center for Astrophysics. Cambridge Mass.

[6] Nelson M.R. The Data Compression Book. Prentice Hall 1991.